Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software UPM ETSIINF.

Examen de Programación II. Convocatoria extraordinaria. 29-6-2015.

Realización: *El test se realizará en la hoja de respuesta. Es* ***importante*** *que no olvidéis rellenar vuestros datos personales y el código clave de vuestro enunciado. Se pueden utilizar hojas aparte en sucio.*

Duración: *La duración total del test será de* ***50 minutos****.*

Puntuación: *El test se valora sobre* ***10 puntos****. Las preguntas tipo test pueden tener una única respuesta o varias respuestas, el enunciado lo deja claro. Cada pregunta con una única respuesta respondida correctamente vale 1 punto, e incorrectamente respondida resta 1/3 puntos. Si en una pregunta con una única respuesta se selecciona más de una respuesta, la pregunta se puntuará con 0 puntos. Para una pregunta con varias respuestas, cada afirmación correcta seleccionada suma* $1/no\_respuestas\_correctas$ *puntos, y cada afirmación incorrecta seleccionada resta* $1/no\_respuestas\_correctas$ *puntos. Las preguntas no contestadas suman 0 puntos en cualquier caso.*

Calificaciones: *Las calificaciones se publicarán en moodle como muy tarde el día* ***1 de Julio de 2015***

Revisión: *Las revisiones serán el día* ***3 de Julio de 2015*** *previa petición por correo electrónico al profesor que haya corregido el examen del alumno.*

Primer Ejercicio

Dada las siguientes definiciones de clase:

```
public class Fecha {
    private int day,month,year;

    public Fecha (int dia, int mes, int anio){
        day=dia;
        month=mes;
        year=anio;
    }

    public Fecha (Fecha f){
        day=f.day;
        month=f.month;
        year=f.year;
    }

    public void setDay(int dia){
            day = dia;
    }

    public  boolean eslgual(Fecha o){
        return day==f.day && month==f.month
           && year==f.year ;
    }
}
```

Pregunta 1

Indicar cuál es la salida por consola del siguiente código.

```
public class Prueba {
  public static void main (String args[]){
    Fecha fecha1 = new Fecha(7,11,2012);
    Fecha fecha2 = new Fecha(fecha1);
    Fecha fecha3 = fecha2;
    fecha3.setDay(10);
    System.out.print(fecha1.eslgual(fecha2)
      + "␣");
    System.out.print(fecha1.eslgual(fecha3)
      + "␣");
    System.out.print((fecha1==fecha2)
      + "␣");
    System.out.print((fecha2==fecha3) );
  }
}
```

Solo una respuesta es correcta.

a) true false false true

b) El código no compila porque se intenta acceder (en el constructor de copia de la Fecha) a atributos de tipo private

c) false false false true

d) false false false false

Segundo Ejercicio

Dada la siguiente clase Division y suponiendo que está definida la excepción ExDivisionPorCero:

```
public class Division{
  public int division(int dividendo,
   int divisor) throws ExDivisionPorCero {
      if(divisor == 0)
         throw new ExDivisionPorCero();
      return dividendo/divisor;
  }
}
```

Indicar cuál es la salida por consola del siguiente código.

```
public class EjercicioDivision {
  public static void main(String[] args) {
    Division div = new Division();
    try {
        int resultado = div.division(1,0);
        System.out.print(resultado);
        int resultado2 = div.division(4,2);
        System.out.print(resultado2);
    } catch (ExDivisionPorCero e) {
        System.out.print("Division␣por␣cero");
    }
    System.out.print("FIN");
  }
}
```

Pregunta 1

Solo una respuesta es correcta.

a) 2FIN

b) 2Exception in thread "main" java.lang.ArithmeticException: / by zero

c) Division por ceroFIN

d) 2Division por ceroFIN

Continúa...

Tercer Ejercicio

Dada la siguiente implementación del método esIgual de una cadena de enteros:

```
public boolean esIgual(CadenaEnteros cadena){
   boolean sigue = true;
   NodoEntero auxCad1= this.cabeza;
   NodoEntero auxCad2= cadena.cabeza;
   while (sigue && auxCad1.getSiguiente()!=null && auxCad2.getSiguiente()!=null)
   {//W
      sigue = auxCad1.getDato() == auxCad2.getDato(); // se comparan los datos
      auxCad1=auxCad1.getSiguiente();
      auxCad2=auxCad2.getSiguiente();
   }//W
   return sigue && auxCad1.getSiguiente()==null && auxCad2.getSiguiente()==null;
}
```

Pregunta 1

Se dice que una implementación falla si no proporciona el resultado correcto. Señala todas las afirmaciones verdaderas. **Puede haber más de una respuesta correcta**.

a) La implementación falla si compara $< 1, 2 >$ y $< 1, 2, 4 >$

b) La implementación falla si compara $< 1, 2, 3 >$ y $< 1, 2, 3 >$

c) La implementación falla si compara $< 1, 2, 3 >$ y $< 1, 2, 4 >$

d) La implementación falla si la cadena o this son cadenas vacías

---

Cuarto Ejercicio

Dada las siguientes definiciones de clase y suponiendo que compilan correctamente:

```
public abstract class Herramienta {
    private double precio;

    public Herramienta (double precio){
        this.precio=precio;
    }

    public double getPrecio(){
        return precio;
    }

    public abstract String accion();
}

public class Martillo extends Herramienta{
    public Martillo(double precio) {
        super(precio);
    }

    public String accion(){
        return "Golpear";
    }
}

public class Destornillador
  extends Herramienta{
    private String tipoPunta;

    public Destornillador(double precio,
      String tipo){
        super(precio);
        tipoPunta = tipo;
    }

    public String accion(){
        return "Atornillar";
    }

    public String getTipoPunta(){
        return tipoPunta;
    }
}
```

Pregunta 1

Dado el siguiente código:

```
public class Prueba {
  public static void main (String args[]){
    Destornillador objeto1 =
       new Destornillador(5,"Estrella");

    Herramienta objeto2 = objeto1;

    Herramienta objeto3 = new Martillo(4);

    Martillo objeto4 = (Martillo)objeto2;

    System.out.print(objeto1.getPrecio()
      + "␣");

    System.out.print(objeto2.getTipoPunta()
      + "␣");

    System.out.print(objeto4.accion() );
  }
}//class
```

Solo una respuesta es correcta.

a) *Martillo objeto*4 = (*Martillo*)*objeto*2; provoca una excepción en tiempo de ejecución.

b) *Martillo objeto*4 = (*Martillo*)*objeto*2; provoca un error en tiempo de compilación.

c) *Herramienta objeto*3 = *new Martillo*(4); provoca una excepción en tiempo de ejecución.

d) *Herramienta objeto*2 = *objeto*1; provoca un error en tiempo de compilación.

Continúa...

Quinto Ejercicio

Dada la siguiente clase:

```
public class A {
 private String nombre;
 private static int cont = 0;
 private static String
     ultimoCreado = "nadie";
 private static String
     ultimoFinalizado = "nadie";

 public A(String nombre) {
   this.nombre = nombre;
   ultimoCreado = nombre;
   cont++;
 }
 public void finalizar() {
   cont--;
   ultimoFinalizado = nombre;
 }
 public static String getEstadistica(){
   return cont + "-" + ultimoCreado + "-"
     + ultimoFinalizado;
 }
}
```

Y la ejecución del siguiente main(), indica cuál será la salida por consola:

```
public static void main(String[] args) {
  A a1, a2;
  a1 = new A("jose");
  a2 = new A("luis");
  System.out.print(A.getEstadistica() + "␣");
  a2.finalizar();
  System.out.print(A.getEstadistica() + "␣");
}
```

Pregunta 1

Indicar cuál de las siguientes afirmaciones es correcta (**Solo una respuesta es correcta**)

a) 2-luis-nadie 1-jose-luis

b) 1-luis-nadie 1-luis-luis

c) 2-luis-nadie 1-luis-luis

d) 2-jose-nadie 1-luis-luis

Sexto Ejercicio

Dadas las siguientes clases:

```
public class Persona {
    private int edad;
    public Persona(int edad){
        this.edad = edad;
    }
    public int getEdad(){
        return edad;
    }
}

public class A {
  private Persona vector[];
  private int posicionesOcupadas;

  public A(Persona v[], int ocupadas){
    vector = v;
    posicionesOcupadas = ocupadas;
  }
  public void incognita(Persona persona){
    boolean sigue = true;
    int i = 0;
    while (sigue){
     if (i == posicionesOcupadas ||
      vector[i].getEdad() > persona.getEdad())
         sigue = false;
     else
         i++;
    }
    vector[i] = persona;
    if (i == posicionesOcupadas)
     posicionesOcupadas++;
  }
  public String toString(){
    String salida = "";
    for(int i=0; i < posicionesOcupadas; i++)
      salida += vector[i].getEdad() + "␣";
    return salida;
  }
}
```

Y la ejecución del siguiente main(), suponiendo que el código compila y se ejecuta correctamente, indica cuál será la salida por consola:

```
public static void main(String[] args) {
    Persona v[] = {new Persona(1),
     new Persona(10), new Persona(15),
      null, null};
    A a = new A(v, 3);
    a.incognita(new Persona(16));
    a.incognita(new Persona(5));
    a.incognita(new Persona(5));
    System.out.print(a);
}
```

Pregunta 1

Indicar cuál de las siguientes afirmaciones es correcta (**Solo una respuesta es correcta**)

a) 1 5 15 16

b) 1 5 5 16 null

c) 1 5 5

d) 1 5 5 16

Continúa...

Séptimo Ejercicio

Respecto al mecanismo de la herencia en Java:

Pregunta 1

Señalar las afirmaciones verdaderas. **Puede haber más de una respuesta correcta**.

a) Java permite herencia múltiple de clases (una clase hija puede tener varias clases padre).

b) Todos los objetos (instancias) de una clase hija son objetos también de la clase padre.

c) Una clase padre solo hereda los métodos públicos de sus clases hijas.

d) Una clase padre puede tener más de una clase hija.

Octavo Ejercicio

Sean las siguientes afirmaciones sobre constructores.

Pregunta 1

Indica cuál de las siguientes afirmaciones es correcta (**Solo una respuesta es correcta**).

a) Es obligatorio definir un método constructor en cualquier clase java, porque java no proporciona uno por defecto

b) El constructor se ejecuta cuando se crea un objeto mediante la sentencia *new*

c) Solo puede haber un método constructor en una clase

d) El constructor siempre debe devolver un resultado mediante la sentencia *return*

Noveno Ejercicio

Dada la implementación de la clase NodoEntero vista en clase:

```
public class NodoEntero {//Nodo
  private NodoEntero siguiente;
  private int dato;
  public NodoEntero(int dato, NodoEntero  siguiente) {
    this.dato=dato;
    this.siguiente=siguiente;
  }
  public NodoEntero  getSiguiente () {
    return this.siguiente;
  }
  public int getDato() {
    return this.dato;
  }
  public void setSiguiente (NodoEntero siguiente) {
    this.siguiente=siguiente;
  }
}//Nodo
```

y el siguiente código:

```
NodoEntero primero = new NodoEntero(5, new NodoEntero(4, null));
NodoEntero aux = primero.getSiguiente();
aux.setSiguiente(new NodoEntero(3,new NodoEntero(1, null)));
```

Pregunta 1

Indicar cuál sería la secuencia de elementos incluida en la cadena enlazada que se crea al ejecutar el código anterior. (**Solo una respuesta es correcta**):

a) [5, 3, 1, 4]

b) [5, 3, 1]

c) [5, 4, 3, 1]

d) [3, 1, 5, 4]

Décimo Ejercicio

Sean las siguiente afirmaciones sobre excepciones:

Pregunta 1

Indicar cuáles de las siguientes afirmaciones son ciertas. (**Puede haber más de una respuesta correcta**.):

a) Las excepciones en java son objetos.

b) La función main() no puede lanzar excepciones.

c) Las excepciones sirven para notificar situaciones anómalas que se presentan durante la ejecución de un programa.

d) En Java sólo se pueden lanzar excepciones predefinidas, es decir, no podemos definir nuevas excepciones.

Fin.